Voice Digit Recognition Using Wavelets

By Michael Del Rose
For Vetronics Technical Center – Technical Paper

Report Documentation Page			*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **04 DEC 2004**	2. REPORT TYPE **Technical Report**	3. DATES COVERED **23-03-2004 to 21-11-2004**	

4. TITLE AND SUBTITLE **Voice Digit Recognition Using Wavelets**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Michael Del Rose**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army TARDEC,6501 East Eleven Mile Rd,Warren,Mi,48397-5000**	8. PERFORMING ORGANIZATION REPORT NUMBER **#14236**

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **U.S. Army TARDEC, 6501 East Eleven Mile Rd, Warren, Mi, 48397-5000**	10. SPONSOR/MONITOR'S ACRONYM(S) **TARDEC**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **#14236**

12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Wavelets are a familiar part of many types of digital signal processing, including compression, audio enhancement, image analysis, and voice recognition. They are an alternative method to Fourier Transforms. The purpose of this paper is to give the reader a sense of how a wavelet is used in voice recognition and what are the results. It will briefly describe what a wavelet is and how to use them in filtering. This paper will not go into great detail on the theory behind wavelets. For a more narrative approach to wavelets please refer to one of the references listed in section 6.1.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT **Public Release**	18. NUMBER OF PAGES **23**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

1.0 Introduction

Wavelets are a familiar part of many types of digital signal processing, including compression, audio enhancement, image analysis, and voice recognition. They are an alternative method to Fourier Transforms. The purpose of this paper is to give the reader a sense of how a wavelet is used in voice recognition and what are the results. It will briefly describe what a wavelet is and how to use them in filtering. This paper will not go into great detail on the theory behind wavelets. For a more narrative approach to wavelets please refer to one of the references listed in section 6.1.

2.0 Wavelets

Wavelets are another tool used in place of Fourier Transforms. The main purpose of the wavelet is not the wavelet itself, but the data that is extracted using the wavelet. The extracted data can be considered the features of the data. The features give a condensed look at the description of the data. For example, when looking at voice recognition, a single spoken digit can be anywhere from 7,000 samples to 24,000 samples (depending on how long the spoken digit is carried out). The purpose of using wavelets in this situation is to cut the data down to a more manageable set, say 10 points transformed from the original data that describes it. These 10 points are the features (hopefully) of the data. If these 10 points are not a good representation of the features of the original data then some sort of manipulation must be performed on them (like taking standard deviations are breaking up into groups, etc.).

Cutting down the data set is not the only thing that the wavelets are tasked to do. When extracting the features from the data, it is the wavelets intent to find information in the data that cannot be seen by normal methods. This may come out of the transformations or it may come from manipulation of the transformed data.

For a transformation function to be considered a wavelet, two things must be true.

1.) The function must integrate to zero:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0$$

2.) The function must have finite energy:

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty$$

The function $\psi(t)$ is called the Mother wavelet. The Mother wavelet is like the starting point for data manipulation. For each level, each new wavelet is taken from an offspring of the Mother wavelet.

2.1 Wavelets vs. Fourier Transforms

The differences in Fourier transforms and wavelet transforms are shown in the common Time-Frequency Squares (see figure 2.1). Notice that the change in the frequency (ω) and time (t) for the Fourier decomposition (figure 2.1a) is fixed. In the wavelet decomposition (figure 2.1b) the changes are not fixed. In fact, the wavelet function is represented in equation 2.1.

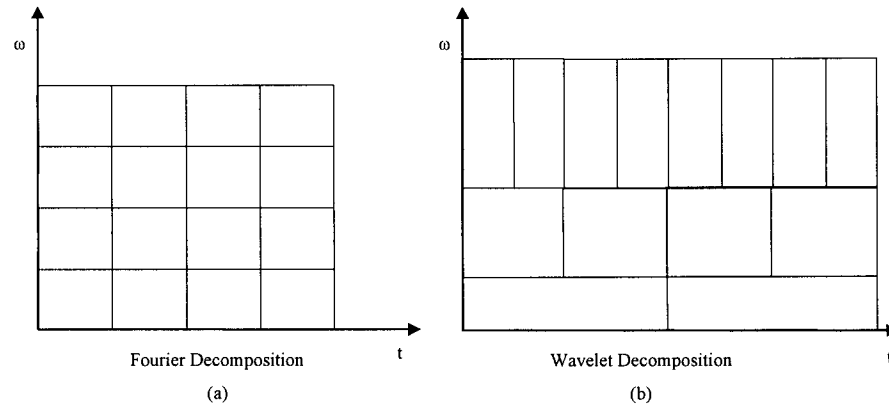$$\psi_{jk} = \psi(2^j t - k) \qquad (eq2.1)$$

Fourier Decomposition

(a)

ω

Wavelet Decomposition

(b)

ω

t

t

Figure 2.1

Equation 2.1 states that the wavelet is compressed by j and shifted k times. Figure 2.1b shows us this. As the frequency increases, the change in time will decrease. So, for higher frequencies, a small time window is need. For lower frequencies, a larger time window is needed. One more note, the area of each box in figure 2.1b (as well as 2.1a) is the same.

2.2 Using Wavelets

When using a wavelet, you can think of it as a filter which extracts the approximation of the data and the detail of the data. At each level, it is important to downsample your data (usually be a factor of 2). This will reduce sample size. You want to keep an even number of samples at each level.

In general, the approximation part of the wavelet filter computes averages. The detail part of the filter computes differences. The tree structure for the wavelet filter looks like figure 2.2.
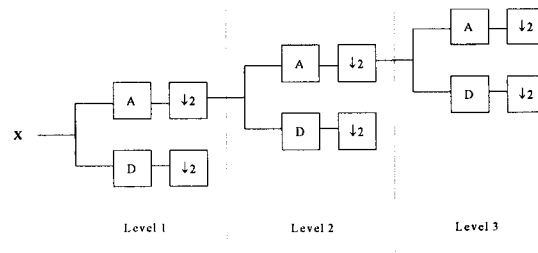
X

A ↓2
D ↓2

A ↓2
D ↓2

A ↓2
D ↓2

Level 1   Level 2   Level 3

Figure 2.2

So, if your data size was 1024, then at level 1 you compute the approximation matrix (A) and the detail matrix (D) using the wavelet. You then decrement this transformed data by 2 (reducing it to 512). Next,

3

you compute the new approximation and detail matrices using the wavelet on the new data, etc., until you are at the level that you want.

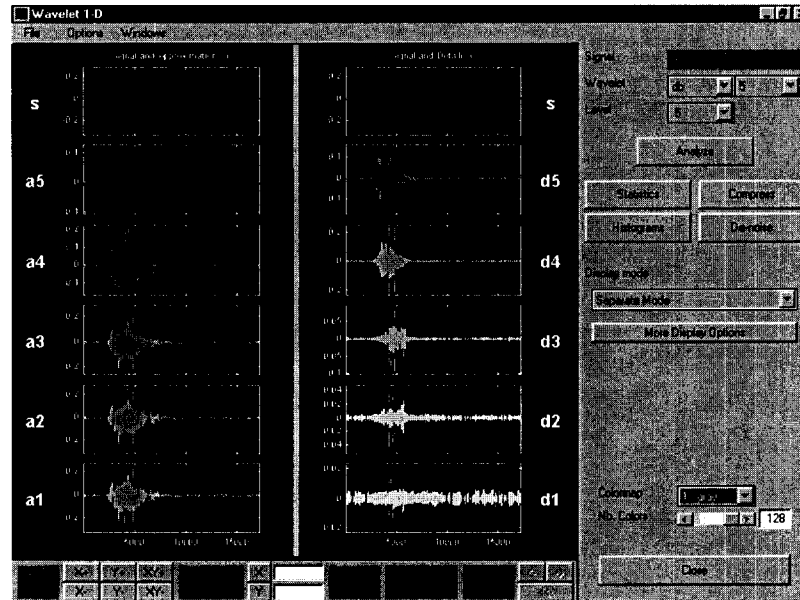If you plot each level out you will have something that looks like figure 2.3.

Figure 2.3

The left side of the Matlab screen (figure 2.3) has the original signal on top and the approximations of the signal at each level below it. The plots to the right contain the original signal on top and the detail plots of each approximation matrix at each level below.

3.1 Voice Recognition of Digits
Spoken voices of digits can take up from 7,000 to 24,000 samples. For a recognition technique to be useful it must make a much smaller number of samples that represent the digit. This new set is called the feature set (or feature vector). From the feature set, a recognition algorithm can be used to classify it. I chose to use the standard deviations of each level of the wavelet transform. This is because the standard deviation is a good representation of the energy of the signal. Continually taking more and more levels of the wavelet is not enough for a feature set.

The data used is the digits from 0 to 9 spoken by two people. I have 21 samples of each digit where I use a combination of them for training and/or testing (depending on the size of the feature vector). The training set had to be at least as large as the number of features in the feature set. If there where 10 features, but I only used 6 sets to train it, then a would get singular matrices. I used Daubechies wavelet to do all the transforming.

The standard deviations are taken of both the approximation and detail coefficient matrices; these are my features, which makes up my feature set. Appendix A shows the Matlab printouts of my program for different levels of both the approximation and detail coefficient matrices. The matrix is read as the classification (column) of the digit (row). For example, the first matrix has 10 training samples and 11 testing samples of each digit (21 total samples - 10 training samples = 11 testing samples). It goes down to level 3. So, 6 features are used as my feature set, standard deviations of the detail matrices at all 3 levels

4

(CD = 3), and the standard deviations of the approximation matrices for all 3 levels (CA = 3). The matrix is read as the classification (column) of the digit (row). So, for the 1 digit, none of them where classified correctly (row 1, column 1) and 10 of them where classified as a 4 (row 1 column 4). The tenth row and column represent 0 (see the first matrix of appendix A).

3.2 Results
Looking at appendix A, I have calculated 5 different classification matrices. Notice that for a 3 level computation with 10 training samples of the digits each (the first matrix) only 40 out of 110 digits were correctly classified (36%). If I increased the training set to all 21 samples each and tested it on the same set I get an improvement of 131 out of 210 correct (62%). This is shown in the third matrix.

For level 5 data, 15 samples each used for training gives a result of 36 out of 60 correct (60%). Using all 21 samples for each digit and testing on the same set gives 199 out of 210 samples classified correctly (95%). See matrices 2 and 4 in appendix A.

The final calculation is based on using 10 levels. Since there are 20 features I had to use each sample in the training set. The testing was done on the same set. The results for this are seen in the final matrix of appendix A. All values were classified correctly.

4.1 Matlab Program
The programs that I used are attached as appendix B and C. Appendix B is used to read in the .wav files that hold the spoken digits and perform the wavelet transform on them. It then saves the relevant information in a file for each class of digits.

Appendix C shows the program that reads in the matrices that hold the information needed. It then performs a Bayes linear classification test on each digit versus the other ones (one at a time). Results are stored in the matrix shown in Appendix A.

5.1 Discussion
Classification was done using Bayes linear classifier. I used this because I am assuming that 21 samples of two people speaking the same word will follow a normal distribution (for those 2 people, only). This may not be true based on different results for the same feature set training on different sample sizes.

Using the standard deviation for the entire wavelet transformed data may also have some consequences. Since speech is varying length, it might have been better to break up each level into groups and taking the standard deviation of each group. If I were to use 4 groups for each level, then my feature vector would increase by a factor of 8 (4 groups for the approximation matrix and 4 groups for the detail matrix). My small sample size would not be able to handle this. If, in the future, more samples are collected then this would be a better option.

I believe I have shown a reasonable method for using wavelets in the classification of spoken digits. Besides the possible wrong assumption of a normal distribution, the data has shown that using the standard deviations of the approximation coefficient and the detail coefficient matrices makes a good feature vector. A level of 5 or greater should be used to get correct classification in the middle to upper nineties.

6.1 References
- G. Strang and T Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press (1997)
- R. Rao and A. Bopardikar, *Wavelet Transforms*, Addison Wesley Longman Inc. (1998)
- B. Hubbard, *The World According to Wavelets*, A K Peters Ltd. (1998)
- K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press (1990)
- J. Harris and H. Stocker, *Handbook of Mathematics and Computational Science*, Springer-Verlag New York Inc. (1998)

Appendix A

********** training samples = 10. CD = 3. CA = 3**********
classMatrix =

0	1	0	10	0	0	0	0	0	0
0	6	0	5	0	0	0	0	0	0
1	2	2	2	1	0	0	0	3	0
1	0	0	9	1	0	0	0	0	0
2	1	0	5	0	0	0	0	3	0
0	0	1	0	0	7	2	0	0	1
1	1	2	1	0	3	3	0	0	0
0	0	2	0	1	0	0	4	4	0
1	0	0	1	1	0	0	0	7	1
2	2	1	2	1	1	0	0	0	2

Matrix 1

********** The total number correct = 40 out of 110. **********

********** training samples = 15. CD = 5. CA = 5**********
classMatrix =

3	0	0	1	2	0	0	0	0	0
1	5	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	5	0	0
0	0	0	6	0	0	0	0	0	0
1	0	0	3	2	0	0	0	0	0
0	0	0	0	0	5	1	0	0	0
1	1	2	0	0	1	0	1	0	0
0	0	0	0	0	0	0	6	0	0
0	0	0	0	0	0	0	0	4	2
0	0	0	0	0	1	0	0	0	5

Matrix 2

********** The total number correct = 36 out of 60. **********

********** training samples = 21. CD = 3. CA = 3**********
classMatrix =

14	0	0	5	2	0	0	0	0	0
1	14	0	5	0	0	1	0	0	0
0	0	9	0	0	0	1	2	9	0
5	0	0	16	0	0	0	0	0	0
12	1	2	2	4	0	0	0	0	0
0	0	0	0	0	18	2	0	0	1
1	1	0	1	0	0	17	0	0	1
0	0	7	0	1	1	0	11	1	0
5	0	0	0	0	0	0	0	16	0
3	2	2	1	0	0	0	1	0	12

Matrix 3

********** The total number correct = 131 out of 210. **********

********** training samples = 21. CD = 5. CA = 5**********
classMatrix =

```
20   0   0   1   0   0   0   0   0   0
 0  21   0   0   0   0   0   0   0   0
 0   0  21   0   0   0   0   0   0   0
 0   0   0  21   0   0   0   0   0   0
 4   0   0   2  15   0   0   0   0   0
 0   0   0   0   0  21   0   0   0   0
 0   0   0   0   0   0  21   0   0   0
 0   0   1   0   0   0   0  19   1   0
 0   0   0   0   0   0   0   0  20   1
 0   0   0   0   0   0   0   0   1  20
```

Matrix 4

********** The total number correct = 199 out of 210. **********

********** training samples = 21. CD = 10. CA = 10**********
classMatrix =

```
21   0   0   0   0   0   0   0   0   0
 0  21   0   0   0   0   0   0   0   0
 0   0  21   0   0   0   0   0   0   0
 0   0   0  21   0   0   0   0   0   0
 0   0   0   0  21   0   0   0   0   0
 0   0   0   0   0  21   0   0   0   0
 0   0   0   0   0   0  21   0   0   0
 0   0   0   0   0   0   0  21   0   0
 0   0   0   0   0   0   0   0  21   0
 0   0   0   0   0   0   0   0   0  21
```

Matrix 5

********** The total number correct = 210 out of 210. **********

Appendix B

```matlab
% matlab code for wavelets feature extractor and saving them into a file
% by mike Del Rose

% assign global variables
WaveletName = 'db4';
WaveletLevel = 10;
NumOfSamples = 21;
SoundNumber = 'one';
SampleNumber = 1;
LastSampleNum = 3;
EndSampleSize = 10000;  % after resampling, we want all matricies to have this size.

cd ..\voice_files

for i=1:1:NumOfSamples
% ***************  1  ***************
   FileName=[ 'one_' num2str(i) '.wav'];
   [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
   sizeSampleY = size(sampleY);
   Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

   LastCA = Y;
   LastCD = Y;
   stdCA=[];
   stdCD=[];
   meanCA=[];
   meanCD=[];
% level 1 - 10
   for ii = 1:1:10
      [CA,dummy] = dwtper(LastCA,WaveletName);
      [dummy,CD] = dwtper(LastCD,WaveletName);
      stdCA = [stdCA,std(CA')];
      meanCA = [meanCA,mean(CA')];
      stdCD = [stdCD,std(CD')];
      meanCD = [meanCD,mean(CD')];
      LastCA = CA;
      LastCD = CD;
   end;  % loop through ii

   stdCA1(i,:) = stdCA;
   stdCD1(i,:) = stdCD;
   meanCA1(i,:) = meanCA;
   meanCD1(i,:) = meanCD;

   clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% ***************  2  ***************
   FileName=[ 'two_' num2str(i) '.wav'];
   [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
   sizeSampleY = size(sampleY);
   Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

   LastCA = Y;
   LastCD = Y;
   stdCA=[];
   stdCD=[];
   meanCA=[];
   meanCD=[];
% level 1 - 10
   for ii = 1:1:10
      [CA,dummy] = dwtper(LastCA,WaveletName);
      [dummy,CD] = dwtper(LastCD,WaveletName);
      stdCA = [stdCA,std(CA')];
      meanCA = [meanCA,mean(CA')];
      stdCD = [stdCD,std(CD')];
      meanCD = [meanCD,mean(CD')];
      LastCA = CA;
```

```
    LastCD = CD;
  end;  % loop through ii

  stdCA2(i,:) = stdCA;
  stdCD2(i,:) = stdCD;
  meanCA2(i,:) = meanCA;
  meanCD2(i,:) = meanCD;

  clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 3  ***************
  FileName=[ 'three_' num2str(i) '.wav'];
  [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
  sizeSampleY = size(sampleY);
  Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

  LastCA = Y;
  LastCD = Y;
  stdCA=[];
  stdCD=[];
  meanCA=[];
  meanCD=[];
% level 1 - 10
  for ii = 1:1:10
    [CA,dummy] = dwtper(LastCA,WaveletName);
    [dummy,CD] = dwtper(LastCD,WaveletName);
    stdCA = [stdCA,std(CA')];
    meanCA = [meanCA,mean(CA')];
    stdCD = [stdCD,std(CD')];
    meanCD = [meanCD,mean(CD')];
    LastCA = CA;
    LastCD = CD;
  end;  % loop through ii

  stdCA3(i,:) = stdCA;
  stdCD3(i,:) = stdCD;
  meanCA3(i,:) = meanCA;
  meanCD3(i,:) = meanCD;

  clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 4  ***************
  FileName=[ 'four_' num2str(i) '.wav'];
  [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
  sizeSampleY = size(sampleY);
  Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

  LastCA = Y;
  LastCD = Y;
  stdCA=[];
  stdCD=[];
  meanCA=[];
  meanCD=[];
% level 1 - 10
  for ii = 1:1:10
    [CA,dummy] = dwtper(LastCA,WaveletName);
    [dummy,CD] = dwtper(LastCD,WaveletName);
    stdCA = [stdCA,std(CA')];
    meanCA = [meanCA,mean(CA')];
    stdCD = [stdCD,std(CD')];
    meanCD = [meanCD,mean(CD')];
    LastCA = CA;
    LastCD = CD;
  end;  % loop through ii

  stdCA4(i,:) = stdCA;
  stdCD4(i,:) = stdCD;
  meanCA4(i,:) = meanCA;
  meanCD4(i,:) = meanCD;
```

```matlab
clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 5 ***************
  FileName=[ 'five_' num2str(i) '.wav'];
  [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
  sizeSampleY = size(sampleY);
  Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

  LastCA = Y;
  LastCD = Y;
  stdCA=[];
  stdCD=[];
  meanCA=[];
  meanCD=[];
% level 1 - 10
  for ii = 1:1:10
    [CA,dummy] = dwtper(LastCA,WaveletName);
    [dummy,CD] = dwtper(LastCD,WaveletName);
    stdCA = [stdCA,std(CA')];
    meanCA = [meanCA,mean(CA')];
    stdCD = [stdCD,std(CD')];
    meanCD = [meanCD,mean(CD')];
    LastCA = CA;
    LastCD = CD;
  end;  % loop through ii

  stdCA5(i,:) = stdCA;
  stdCD5(i,:) = stdCD;
  meanCA5(i,:) = meanCA;
  meanCD5(i,:) = meanCD;

  clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 6 ***************
  FileName=[ 'six_' num2str(i) '.wav'];
  [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
  sizeSampleY = size(sampleY);
  Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

  LastCA = Y;
  LastCD = Y;
  stdCA=[];
  stdCD=[];
  meanCA=[];
  meanCD=[];
% level 1 - 10
  for ii = 1:1:10
    [CA,dummy] = dwtper(LastCA,WaveletName);
    [dummy,CD] = dwtper(LastCD,WaveletName);
    stdCA = [stdCA,std(CA')];
    meanCA = [meanCA,mean(CA')];
    stdCD = [stdCD,std(CD')];
    meanCD = [meanCD,mean(CD')];
    LastCA = CA;
    LastCD = CD;
  end;  % loop through ii

  stdCA6(i,:) = stdCA;
  stdCD6(i,:) = stdCD;
  meanCA6(i,:) = meanCA;
  meanCD6(i,:) = meanCD;

  clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 7 ***************
  FileName=[ 'seven_' num2str(i) '.wav'];
  [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
  sizeSampleY = size(sampleY);
  Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size
```

```matlab
   LastCA = Y;
   LastCD = Y;
   stdCA=[];
   stdCD=[];
   meanCA=[];
   meanCD=[];
% level 1 - 10
   for ii = 1:1:10
      [CA,dummy] = dwtper(LastCA,WaveletName);
      [dummy,CD] = dwtper(LastCD,WaveletName);
      stdCA = [stdCA,std(CA')];
      meanCA = [meanCA,mean(CA')];
      stdCD = [stdCD,std(CD')];
      meanCD = [meanCD,mean(CD')];
      LastCA = CA;
      LastCD = CD;
   end; % loop through ii

   stdCA7(i,:) = stdCA;
   stdCD7(i,:) = stdCD;
   meanCA7(i,:) = meanCA;
   meanCD7(i,:) = meanCD;

   clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 8 ***************
   FileName=[ 'eight_' num2str(i) '.wav'];
   [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
   sizeSampleY = size(sampleY);
   Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

   LastCA = Y;
   LastCD = Y;
   stdCA=[];
   stdCD=[];
   meanCA=[];
   meanCD=[];
% level 1 - 10
   for ii = 1:1:10
      [CA,dummy] = dwtper(LastCA,WaveletName);
      [dummy,CD] = dwtper(LastCD,WaveletName);
      stdCA = [stdCA,std(CA')];
      meanCA = [meanCA,mean(CA')];
      stdCD = [stdCD,std(CD')];
      meanCD = [meanCD,mean(CD')];
      LastCA = CA;
      LastCD = CD;
   end; % loop through ii

   stdCA8(i,:) = stdCA;
   stdCD8(i,:) = stdCD;
   meanCA8(i,:) = meanCA;
   meanCD8(i,:) = meanCD;

   clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 9 ***************
   FileName=[ 'nine_' num2str(i) '.wav'];
   [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
   sizeSampleY = size(sampleY);
   Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

   LastCA = Y;
   LastCD = Y;
   stdCA=[];
   stdCD=[];
   meanCA=[];
   meanCD=[];
% level 1 - 10
   for ii = 1:1:10
```

```matlab
    [CA,dummy] = dwtper(LastCA,WaveletName);
    [dummy,CD] = dwtper(LastCD,WaveletName);
    stdCA = [stdCA,std(CA')];
    meanCA = [meanCA,mean(CA')];
    stdCD = [stdCD,std(CD')];
    meanCD = [meanCD,mean(CD')];
    LastCA = CA;
    LastCD = CD;
  end;  % loop through ii

  stdCA9(i,:) = stdCA;
  stdCD9(i,:) = stdCD;
  meanCA9(i,:) = meanCA;
  meanCD9(i,:) = meanCD;

  clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

% *************** 0 ***************
  FileName=[ 'zero_' num2str(i) '.wav'];
  [sampleY,FS,Bits] = wavread(FileName);  % read in the .wav file and assign signal to Y
  sizeSampleY = size(sampleY);
  Y = resample(sampleY, EndSampleSize, sizeSampleY(1));  % need to make all matrices the same size

  LastCA = Y;
  LastCD = Y;
  stdCA=[];
  stdCD=[];
  meanCA=[];
  meanCD=[];
% level 1 - 10
  for ii = 1:1:10
    [CA,dummy] = dwtper(LastCA,WaveletName);
    [dummy,CD] = dwtper(LastCD,WaveletName);
    stdCA = [stdCA,std(CA')];
    meanCA = [meanCA,mean(CA')];
    stdCD = [stdCD,std(CD')];
    meanCD = [meanCD,mean(CD')];
    LastCA = CA;
    LastCD = CD;
  end;  % loop through ii

  stdCA0(i,:) = stdCA;
  stdCD0(i,:) = stdCD;
  meanCA0(i,:) = meanCA;
  meanCD0(i,:) = meanCD;

  clear CD CA LastCD LastCA stdCD stdCA meanCD meanCA;

end;  % loop through i

save 'stats_1' stdCD1 stdCA1 meanCD1 meanCA1;
save 'stats_2' stdCD2 stdCA2 meanCD2 meanCA2;
save 'stats_3' stdCD3 stdCA3 meanCD3 meanCA3;
save 'stats_4' stdCD4 stdCA4 meanCD4 meanCA4;
save 'stats_5' stdCD5 stdCA5 meanCD5 meanCA5;
save 'stats_6' stdCD6 stdCA6 meanCD6 meanCA6;
save 'stats_7' stdCD7 stdCA7 meanCD7 meanCA7;
save 'stats_8' stdCD8 stdCA8 meanCD8 meanCA8;
save 'stats_9' stdCD9 stdCA9 meanCD9 meanCA9;
save 'stats_0' stdCD0 stdCA0 meanCD0 meanCA0;
```

12

Appendix C

```
% matlab code for wavelets - wave_proj5.m
%
% This program reads in feature matricies.  It then performs
% a Bayes Linear Classifier test.
% by Mike Del Rose

clear all;

% ********** Constants **********
ON = 1;
OFF = 0;

% ********** Debug Variables **********
DEBUG_KEEP = OFF;
DEBUG_GRAPH = OFF;
DEBUG_CODE = OFF;
DebugSignalNum = 1;  % the signal for debug (like voice one ==> DebugSignalNum = 1)
DebugSampleNum = 2;  % the sample number of the signal specified in DebugSignalNum
DEBUG_CLASS_1 = ON;
DEBUG_CLASS_2 = ON;
DEBUG_CLASS_3 = ON;
DEBUG_CLASS_4 = ON;
DEBUG_CLASS_5 = ON;
DEBUG_CLASS_6 = ON;
DEBUG_CLASS_7 = ON;
DEBUG_CLASS_8 = ON;
DEBUG_CLASS_9 = ON;
DEBUG_CLASS_0 = ON;


% ********** Global Variables **********
WaveletName = 'db4';
WaveletLevel = 10;
NumOfSamples = 21;
EndSampleSize = 10000;  % after resampling, we want all matricies to have this size.
NumOfCDfeat = 5;  % number of features taken from the Detail coeff. out of 10
NumOfCAfeat = 1;  % number of features taken from the Approx coeef. out of 10
TotalNumOfFeat = NumOfCDfeat + NumOfCAfeat; % total number of features used.
trainPts = 15;  % number of points used for training.  Testing will be using the remainder points.
if NumOfSamples <= trainPts
   testPts = NumOfSamples;
else
   testPts = NumOfSamples - trainPts;
end;

classMatrix = zeros(10,10);  % holds the classification of the test data in question


cd ..\voice_files

% ****************************************************************
% *************** Read in stats files and process ***************
% ****************************************************************

for i=0:1:9
   FileName=[ 'stats_' num2str(i)];
   eval(['load ' FileName ' stdCD' num2str(i) ' stdCA' num2str(i)]);
end;  % loop through ii

trainA1 = [stdCD1(1:trainPts,1:NumOfCDfeat) stdCA1(1:trainPts,1:NumOfCAfeat)];
trainA2 = [stdCD2(1:trainPts,1:NumOfCDfeat) stdCA2(1:trainPts,1:NumOfCAfeat)];
trainA3 = [stdCD3(1:trainPts,1:NumOfCDfeat) stdCA3(1:trainPts,1:NumOfCAfeat)];
trainA4 = [stdCD4(1:trainPts,1:NumOfCDfeat) stdCA4(1:trainPts,1:NumOfCAfeat)];
trainA5 = [stdCD5(1:trainPts,1:NumOfCDfeat) stdCA5(1:trainPts,1:NumOfCAfeat)];
trainA6 = [stdCD6(1:trainPts,1:NumOfCDfeat) stdCA6(1:trainPts,1:NumOfCAfeat)];
trainA7 = [stdCD7(1:trainPts,1:NumOfCDfeat) stdCA7(1:trainPts,1:NumOfCAfeat)];
trainA8 = [stdCD8(1:trainPts,1:NumOfCDfeat) stdCA8(1:trainPts,1:NumOfCAfeat)];
```

```
trainA9 = [stdCD9(1:trainPts,1:NumOfCDfeat) stdCA9(1:trainPts,1:NumOfCAfeat)];
trainA0 = [stdCD0(1:trainPts,1:NumOfCDfeat) stdCA0(1:trainPts,1:NumOfCAfeat)];
A0 = [stdCD0(:,1:NumOfCDfeat) stdCA0(:,1:NumOfCAfeat)];
A1 = [stdCD1(:,1:NumOfCDfeat) stdCA1(:,1:NumOfCAfeat)];
A2 = [stdCD2(:,1:NumOfCDfeat) stdCA2(:,1:NumOfCAfeat)];
A3 = [stdCD3(:,1:NumOfCDfeat) stdCA3(:,1:NumOfCAfeat)];
A4 = [stdCD4(:,1:NumOfCDfeat) stdCA4(:,1:NumOfCAfeat)];
A5 = [stdCD5(:,1:NumOfCDfeat) stdCA5(:,1:NumOfCAfeat)];
A6 = [stdCD6(:,1:NumOfCDfeat) stdCA6(:,1:NumOfCAfeat)];
A7 = [stdCD7(:,1:NumOfCDfeat) stdCA7(:,1:NumOfCAfeat)];
A8 = [stdCD8(:,1:NumOfCDfeat) stdCA8(:,1:NumOfCAfeat)];
A9 = [stdCD9(:,1:NumOfCDfeat) stdCA9(:,1:NumOfCAfeat)];
A0 = [stdCD0(:,1:NumOfCDfeat) stdCA0(:,1:NumOfCAfeat)];

% ****************************************************************
% *************** Printing graphs on DEBUG_GRAPH ***************
% ****************************************************************

if DEBUG_GRAPH == ON

   for iGraph = 0:1:9
      X = [1:TotalNumOfFeat];
      figure(iGraph+1)
      Ydata = ['A' num2str(iGraph)];
      Y = eval(Ydata);
      plot(X,Y,'b-')
      TitleOutput = ['features for :' num2str(iGraph)];
      title(TitleOutput)
   end;

end;

% ****************************************************************
% **************** Classify the wave features ******************
% ****************************************************************
covA1 = cov(trainA1);
covA2 = cov(trainA2);
covA3 = cov(trainA3);
covA4 = cov(trainA4);
covA5 = cov(trainA5);
covA6 = cov(trainA6);
covA7 = cov(trainA7);
covA8 = cov(trainA8);
covA9 = cov(trainA9);
covA0 = cov(trainA0);
meanA1 = mean(trainA1)';
meanA2 = mean(trainA2)';
meanA3 = mean(trainA3)';
meanA4 = mean(trainA4)';
meanA5 = mean(trainA5)';
meanA6 = mean(trainA6)';
meanA7 = mean(trainA7)';
meanA8 = mean(trainA8)';
meanA9 = mean(trainA9)';
meanA0 = mean(trainA0)';


% ===================================
%    BAYES TEST
% ===================================
% Threshold test.
P1 = .5;
P2 = .5;
log_likelihood = log(P1/P2);

for testSample = NumOfSamples - testPts + 1:1:NumOfSamples

if (DEBUG_CLASS_1)
% Calculate negative log likelihood ratio and test Y1
    part1 = .5 * (A1(testSample,:)' - meanA1)' * inv(covA1) * (A1(testSample,:)' - meanA1);
```

```
    Case1v2 = part1 - ...
        .5 * (A1(testSample,:)' - meanA2)' * inv(covA2) * (A1(testSample,:)' - meanA2) + ...
        .5 * log( (det(covA1))/(det(covA2)) );
    Case1v3 = part1 - ...
        .5 * (A1(testSample,:)' - meanA3)' * inv(covA3) * (A1(testSample,:)' - meanA3) + ...
        .5 * log( (det(covA1))/(det(covA3)) );
    Case1v4 = part1 - ...
        .5 * (A1(testSample,:)' - meanA4)' * inv(covA4) * (A1(testSample,:)' - meanA4) + ...
        .5 * log( (det(covA1))/(det(covA4)) );
    Case1v5 = part1 - ...
        .5 * (A1(testSample,:)' - meanA5)' * inv(covA5) * (A1(testSample,:)' - meanA5) + ...
        .5 * log( (det(covA1))/(det(covA5)) );
    Case1v6 = part1 - ...
        .5 * (A1(testSample,:)' - meanA6)' * inv(covA6) * (A1(testSample,:)' - meanA6) + ...
        .5 * log( (det(covA1))/(det(covA6)) );
    Case1v7 = part1 - ...
        .5 * (A1(testSample,:)' - meanA7)' * inv(covA7) * (A1(testSample,:)' - meanA7) + ...
        .5 * log( (det(covA1))/(det(covA7)) );
    Case1v8 = part1 - ...
        .5 * (A1(testSample,:)' - meanA8)' * inv(covA8) * (A1(testSample,:)' - meanA8) + ...
        .5 * log( (det(covA1))/(det(covA8)) );
    Case1v9 = part1 - ...
        .5 * (A1(testSample,:)' - meanA9)' * inv(covA9) * (A1(testSample,:)' - meanA9) + ...
        .5 * log( (det(covA1))/(det(covA9)) );
    Case1v0 = part1 - ...
        .5 * (A1(testSample,:)' - meanA0)' * inv(covA0) * (A1(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA1))/(det(covA0)) );
end;

if (DEBUG_CLASS_2)
% Check class 2
    part1_2 = .5 * (A2(testSample,:)' - meanA2)' * inv(covA2) * (A2(testSample,:)' - meanA2);
    Case2v1 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA1)' * inv(covA1) * (A2(testSample,:)' - meanA1) + ...
        .5 * log( (det(covA2))/(det(covA1)) );
    Case2v3 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA3)' * inv(covA3) * (A2(testSample,:)' - meanA3) + ...
        .5 * log( (det(covA2))/(det(covA3)) );
    Case2v4 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA4)' * inv(covA4) * (A2(testSample,:)' - meanA4) + ...
        .5 * log( (det(covA2))/(det(covA4)) );
    Case2v5 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA5)' * inv(covA5) * (A2(testSample,:)' - meanA5) + ...
        .5 * log( (det(covA2))/(det(covA5)) );
    Case2v6 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA6)' * inv(covA6) * (A2(testSample,:)' - meanA6) + ...
        .5 * log( (det(covA2))/(det(covA6)) );
    Case2v7 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA7)' * inv(covA7) * (A2(testSample,:)' - meanA7) + ...
        .5 * log( (det(covA2))/(det(covA7)) );
    Case2v8 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA8)' * inv(covA8) * (A2(testSample,:)' - meanA8) + ...
        .5 * log( (det(covA2))/(det(covA8)) );
    Case2v9 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA9)' * inv(covA9) * (A2(testSample,:)' - meanA9) + ...
        .5 * log( (det(covA2))/(det(covA9)) );
    Case2v0 = part1_2 - ...
        .5 * (A2(testSample,:)' - meanA0)' * inv(covA0) * (A2(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA2))/(det(covA0)) );
end;

if (DEBUG_CLASS_3)
% Check class 3
    part1_3 = .5 * (A3(testSample,:)' - meanA3)' * inv(covA3) * (A3(testSample,:)' - meanA3);
    Case3v1 = part1_3 - ...
        .5 * (A3(testSample,:)' - meanA1)' * inv(covA1) * (A3(testSample,:)' - meanA1) + ...
        .5 * log( (det(covA3))/(det(covA1)) );
    Case3v2 = part1_3 - ...
        .5 * (A3(testSample,:)' - meanA2)' * inv(covA2) * (A3(testSample,:)' - meanA2) + ...
        .5 * log( (det(covA3))/(det(covA2)) );
```

15

```
  Case3v4 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA4)' * inv(covA4) * (A3(testSample,:)' - meanA4) + ...
      .5 * log( (det(covA3))/(det(covA4)) );
  Case3v5 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA5)' * inv(covA5) * (A3(testSample,:)' - meanA5) + ...
      .5 * log( (det(covA3))/(det(covA5)) );
  Case3v6 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA6)' * inv(covA6) * (A3(testSample,:)' - meanA6) + ...
      .5 * log( (det(covA3))/(det(covA6)) );
  Case3v7 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA7)' * inv(covA7) * (A3(testSample,:)' - meanA7) + ...
      .5 * log( (det(covA3))/(det(covA7)) );
  Case3v8 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA8)' * inv(covA8) * (A3(testSample,:)' - meanA8) + ...
      .5 * log( (det(covA3))/(det(covA8)) );
  Case3v9 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA9)' * inv(covA9) * (A3(testSample,:)' - meanA9) + ...
      .5 * log( (det(covA3))/(det(covA9)) );
  Case3v0 = part1_3 - ...
      .5 * (A3(testSample,:)' - meanA0)' * inv(covA0) * (A3(testSample,:)' - meanA0) + ...
      .5 * log( (det(covA3))/(det(covA0)) );
end;

if (DEBUG_CLASS_4)
% Check class 4
  part1_4 = .5 * (A4(testSample,:)' - meanA4)' * inv(covA4) * (A4(testSample,:)' - meanA4);
  Case4v1 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA1)' * inv(covA1) * (A4(testSample,:)' - meanA1) + ...
      .5 * log( (det(covA4))/(det(covA1)) );
  Case4v2 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA2)' * inv(covA2) * (A4(testSample,:)' - meanA2) + ...
      .5 * log( (det(covA4))/(det(covA2)) );
  Case4v3 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA3)' * inv(covA3) * (A4(testSample,:)' - meanA3) + ...
      .5 * log( (det(covA4))/(det(covA3)) );
  Case4v5 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA5)' * inv(covA5) * (A4(testSample,:)' - meanA5) + ...
      .5 * log( (det(covA4))/(det(covA5)) );
  Case4v6 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA6)' * inv(covA6) * (A4(testSample,:)' - meanA6) + ...
      .5 * log( (det(covA4))/(det(covA6)) );
  Case4v7 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA7)' * inv(covA7) * (A4(testSample,:)' - meanA7) + ...
      .5 * log( (det(covA4))/(det(covA7)) );
  Case4v8 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA8)' * inv(covA8) * (A4(testSample,:)' - meanA8) + ...
      .5 * log( (det(covA4))/(det(covA8)) );
  Case4v9 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA9)' * inv(covA9) * (A4(testSample,:)' - meanA9) + ...
      .5 * log( (det(covA4))/(det(covA9)) );
  Case4v0 = part1_4 - ...
      .5 * (A4(testSample,:)' - meanA0)' * inv(covA0) * (A4(testSample,:)' - meanA0) + ...
      .5 * log( (det(covA4))/(det(covA0)) );
end;

if (DEBUG_CLASS_5)
% Check class 5
  part1_5 = .5 * (A5(testSample,:)' - meanA5)' * inv(covA5) * (A5(testSample,:)' - meanA5);
  Case5v1 = part1_5 - ...
      .5 * (A5(testSample,:)' - meanA1)' * inv(covA1) * (A5(testSample,:)' - meanA1) + ...
      .5 * log( (det(covA5))/(det(covA1)) );
  Case5v2 = part1_5 - ...
      .5 * (A5(testSample,:)' - meanA2)' * inv(covA2) * (A5(testSample,:)' - meanA2) + ...
      .5 * log( (det(covA5))/(det(covA2)) );
  Case5v3 = part1_5 - ...
      .5 * (A5(testSample,:)' - meanA3)' * inv(covA3) * (A5(testSample,:)' - meanA3) + ...
      .5 * log( (det(covA5))/(det(covA3)) );
  Case5v4 = part1_5 - ...
      .5 * (A5(testSample,:)' - meanA4)' * inv(covA4) * (A5(testSample,:)' - meanA4) + ...
      .5 * log( (det(covA5))/(det(covA4)) );
```

16

```
    Case5v6 = part1_5 - ...
        .5 * (A5(testSample,:)' - meanA6)' * inv(covA6) * (A5(testSample,:)' - meanA6) + ...
        .5 * log( (det(covA5))/(det(covA6)) );
    Case5v7 = part1_5 - ...
        .5 * (A5(testSample,:)' - meanA7)' * inv(covA7) * (A5(testSample,:)' - meanA7) + ...
        .5 * log( (det(covA5))/(det(covA7)) );
    Case5v8 = part1_5 - ...
        .5 * (A5(testSample,:)' - meanA8)' * inv(covA8) * (A5(testSample,:)' - meanA8) + ...
        .5 * log( (det(covA5))/(det(covA8)) );
    Case5v9 = part1_5 - ...
        .5 * (A5(testSample,:)' - meanA9)' * inv(covA9) * (A5(testSample,:)' - meanA9) + ...
        .5 * log( (det(covA5))/(det(covA9)) );
    Case5v0 = part1_5 - ...
        .5 * (A5(testSample,:)' - meanA0)' * inv(covA0) * (A5(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA5))/(det(covA0)) );
end;

if (DEBUG_CLASS_6)
% Check class 6
    part1_6 = .5 * (A6(testSample,:)' - meanA6)' * inv(covA6) * (A6(testSample,:)' - meanA6);
    Case6v1 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA1)' * inv(covA1) * (A6(testSample,:)' - meanA1) + ...
        .5 * log( (det(covA6))/(det(covA1)) );
    Case6v2 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA2)' * inv(covA2) * (A6(testSample,:)' - meanA2) + ...
        .5 * log( (det(covA6))/(det(covA2)) );
    Case6v3 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA3)' * inv(covA3) * (A6(testSample,:)' - meanA3) + ...
        .5 * log( (det(covA6))/(det(covA3)) );
    Case6v4 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA4)' * inv(covA4) * (A6(testSample,:)' - meanA4) + ...
        .5 * log( (det(covA6))/(det(covA4)) );
    Case6v5 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA5)' * inv(covA5) * (A6(testSample,:)' - meanA5) + ...
        .5 * log( (det(covA6))/(det(covA5)) );
    Case6v7 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA7)' * inv(covA7) * (A6(testSample,:)' - meanA7) + ...
        .5 * log( (det(covA6))/(det(covA7)) );
    Case6v8 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA8)' * inv(covA8) * (A6(testSample,:)' - meanA8) + ...
        .5 * log( (det(covA6))/(det(covA8)) );
    Case6v9 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA9)' * inv(covA9) * (A6(testSample,:)' - meanA9) + ...
        .5 * log( (det(covA6))/(det(covA9)) );
    Case6v0 = part1_6 - ...
        .5 * (A6(testSample,:)' - meanA0)' * inv(covA0) * (A6(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA6))/(det(covA0)) );
end;

if (DEBUG_CLASS_7)
% Check class 7
    part1_7 = .5 * (A7(testSample,:)' - meanA7)' * inv(covA7) * (A7(testSample,:)' - meanA7);
    Case7v1 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA1)' * inv(covA1) * (A7(testSample,:)' - meanA1) + ...
        .5 * log( (det(covA7))/(det(covA1)) );
    Case7v2 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA2)' * inv(covA2) * (A7(testSample,:)' - meanA2) + ...
        .5 * log( (det(covA7))/(det(covA2)) );
    Case7v3 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA3)' * inv(covA3) * (A7(testSample,:)' - meanA3) + ...
        .5 * log( (det(covA7))/(det(covA3)) );
    Case7v4 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA4)' * inv(covA4) * (A7(testSample,:)' - meanA4) + ...
        .5 * log( (det(covA7))/(det(covA4)) );
    Case7v5 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA5)' * inv(covA5) * (A7(testSample,:)' - meanA5) + ...
        .5 * log( (det(covA7))/(det(covA5)) );
    Case7v6 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA6)' * inv(covA6) * (A7(testSample,:)' - meanA6) + ...
        .5 * log( (det(covA7))/(det(covA6)) );
```

```
    Case7v8 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA8)' * inv(covA8) * (A7(testSample,:)' - meanA8) + ...
        .5 * log( (det(covA7))/(det(covA8)) );
    Case7v9 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA9)' * inv(covA9) * (A7(testSample,:)' - meanA9) + ...
        .5 * log( (det(covA7))/(det(covA9)) );
    Case7v0 = part1_7 - ...
        .5 * (A7(testSample,:)' - meanA0)' * inv(covA0) * (A7(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA7))/(det(covA0)) );
end;

if (DEBUG_CLASS_8)
% Check class 8
    part1_8 = .5 * (A8(testSample,:)' - meanA8)' * inv(covA8) * (A8(testSample,:)' - meanA8);
    Case8v1 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA1)' * inv(covA1) * (A8(testSample,:)' - meanA1) + ...
        .5 * log( (det(covA8))/(det(covA1)) );
    Case8v2 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA2)' * inv(covA2) * (A8(testSample,:)' - meanA2) + ...
        .5 * log( (det(covA8))/(det(covA2)) );
    Case8v3 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA3)' * inv(covA3) * (A8(testSample,:)' - meanA3) + ...
        .5 * log( (det(covA8))/(det(covA3)) );
    Case8v4 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA4)' * inv(covA4) * (A8(testSample,:)' - meanA4) + ...
        .5 * log( (det(covA8))/(det(covA4)) );
    Case8v5 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA5)' * inv(covA5) * (A8(testSample,:)' - meanA5) + ...
        .5 * log( (det(covA8))/(det(covA5)) );
    Case8v6 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA6)' * inv(covA6) * (A8(testSample,:)' - meanA6) + ...
        .5 * log( (det(covA8))/(det(covA6)) );
    Case8v7 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA7)' * inv(covA7) * (A8(testSample,:)' - meanA7) + ...
        .5 * log( (det(covA8))/(det(covA7)) );
    Case8v9 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA9)' * inv(covA9) * (A8(testSample,:)' - meanA9) + ...
        .5 * log( (det(covA8))/(det(covA9)) );
    Case8v0 = part1_8 - ...
        .5 * (A8(testSample,:)' - meanA0)' * inv(covA0) * (A8(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA8))/(det(covA0)) );
end;

if (DEBUG_CLASS_9)
% Check class 9
    part1_9 = .5 * (A9(testSample,:)' - meanA9)' * inv(covA9) * (A9(testSample,:)' - meanA9);
    Case9v1 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA1)' * inv(covA1) * (A9(testSample,:)' - meanA1) + ...
        .5 * log( (det(covA9))/(det(covA1)) );
    Case9v2 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA2)' * inv(covA2) * (A9(testSample,:)' - meanA2) + ...
        .5 * log( (det(covA9))/(det(covA2)) );
    Case9v3 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA3)' * inv(covA3) * (A9(testSample,:)' - meanA3) + ...
        .5 * log( (det(covA9))/(det(covA3)) );
    Case9v4 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA4)' * inv(covA4) * (A9(testSample,:)' - meanA4) + ...
        .5 * log( (det(covA9))/(det(covA4)) );
    Case9v5 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA5)' * inv(covA5) * (A9(testSample,:)' - meanA5) + ...
        .5 * log( (det(covA9))/(det(covA5)) );
    Case9v6 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA6)' * inv(covA6) * (A9(testSample,:)' - meanA6) + ...
        .5 * log( (det(covA9))/(det(covA6)) );
    Case9v7 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA7)' * inv(covA7) * (A9(testSample,:)' - meanA7) + ...
        .5 * log( (det(covA9))/(det(covA7)) );
    Case9v8 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA8)' * inv(covA8) * (A9(testSample,:)' - meanA8) + ...
        .5 * log( (det(covA9))/(det(covA8)) );
```

```
    Case9v0 = part1_9 - ...
        .5 * (A9(testSample,:)' - meanA0)' * inv(covA0) * (A9(testSample,:)' - meanA0) + ...
        .5 * log( (det(covA9))/(det(covA0)) );
    end;

    if (DEBUG_CLASS_0)
    % Check class 0
        part1_0 = .5 * (A0(testSample,:)' - meanA0)' * inv(covA0) * (A0(testSample,:)' - meanA0);
        Case0v1 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA1)' * inv(covA1) * (A0(testSample,:)' - meanA1) + ...
            .5 * log( (det(covA0))/(det(covA1)) );
        Case0v2 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA2)' * inv(covA2) * (A0(testSample,:)' - meanA2) + ...
            .5 * log( (det(covA0))/(det(covA2)) );
        Case0v3 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA3)' * inv(covA3) * (A0(testSample,:)' - meanA3) + ...
            .5 * log( (det(covA0))/(det(covA3)) );
        Case0v4 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA4)' * inv(covA4) * (A0(testSample,:)' - meanA4) + ...
            .5 * log( (det(covA0))/(det(covA4)) );
        Case0v5 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA5)' * inv(covA5) * (A0(testSample,:)' - meanA5) + ...
            .5 * log( (det(covA0))/(det(covA5)) );
        Case0v6 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA6)' * inv(covA6) * (A0(testSample,:)' - meanA6) + ...
            .5 * log( (det(covA0))/(det(covA6)) );
        Case0v7 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA7)' * inv(covA7) * (A0(testSample,:)' - meanA7) + ...
            .5 * log( (det(covA0))/(det(covA7)) );
        Case0v8 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA8)' * inv(covA8) * (A0(testSample,:)' - meanA8) + ...
            .5 * log( (det(covA0))/(det(covA8)) );
        Case0v9 = part1_0 - ...
            .5 * (A0(testSample,:)' - meanA9)' * inv(covA9) * (A0(testSample,:)' - meanA9) + ...
            .5 * log( (det(covA0))/(det(covA9)) );
    end;
    % check log likelihood against each class.

    if (DEBUG_CLASS_1)
    % Class 1
        if Case1v2 > log_likelihood
            classMatrix(1,2) = classMatrix(1,2) + 1;
        elseif Case1v3 > log_likelihood
            classMatrix(1,3) = classMatrix(1,3) + 1;
        elseif Case1v4 > log_likelihood
            classMatrix(1,4) = classMatrix(1,4) + 1;
        elseif Case1v5 > log_likelihood
            classMatrix(1,5) = classMatrix(1,5) + 1;
        elseif Case1v6 > log_likelihood
            classMatrix(1,6) = classMatrix(1,6) + 1;
        elseif Case1v7 > log_likelihood
            classMatrix(1,7) = classMatrix(1,7) + 1;
        elseif Case1v8 > log_likelihood
            classMatrix(1,8) = classMatrix(1,8) + 1;
        elseif Case1v9 > log_likelihood
            classMatrix(1,9) = classMatrix(1,9) + 1;
        elseif Case1v0 > log_likelihood
            classMatrix(1,10) = classMatrix(1,10) + 1;
        else
            classMatrix(1,1) = classMatrix(1,1) + 1;
        end;
    end;

    if (DEBUG_CLASS_2)
    % Class 2
        if Case2v1 > log_likelihood
            classMatrix(2,1) = classMatrix(2,1) + 1;
        elseif Case2v3 > log_likelihood
            classMatrix(2,3) = classMatrix(2,3) + 1;
        elseif Case2v4 > log_likelihood
```

```
      classMatrix(2,4) = classMatrix(2,4) + 1;
    elseif Case2v5 > log_likelihood
      classMatrix(2,5) = classMatrix(2,5) + 1;
    elseif Case2v6 > log_likelihood
      classMatrix(2,6) = classMatrix(2,6) + 1;
    elseif Case2v7 > log_likelihood
      classMatrix(2,7) = classMatrix(2,7) + 1;
    elseif Case2v8 > log_likelihood
      classMatrix(2,8) = classMatrix(2,8) + 1;
    elseif Case2v9 > log_likelihood
      classMatrix(2,9) = classMatrix(2,9) + 1;
    elseif Case2v0 > log_likelihood
      classMatrix(2,10) = classMatrix(2,10) + 1;
    else
      classMatrix(2,2) = classMatrix(2,2) + 1;
    end;
  end;

  if (DEBUG_CLASS_3)
  % Class 3
    if Case3v1 > log_likelihood
      classMatrix(3,1) = classMatrix(3,1) + 1;
    elseif Case3v2 > log_likelihood
      classMatrix(3,2) = classMatrix(3,2) + 1;
    elseif Case3v4 > log_likelihood
      classMatrix(3,4) = classMatrix(3,4) + 1;
    elseif Case3v5 > log_likelihood
      classMatrix(3,5) = classMatrix(3,5) + 1;
    elseif Case3v6 > log_likelihood
      classMatrix(3,6) = classMatrix(3,6) + 1;
    elseif Case3v7 > log_likelihood
      classMatrix(3,7) = classMatrix(3,7) + 1;
    elseif Case3v8 > log_likelihood
      classMatrix(3,8) = classMatrix(3,8) + 1;
    elseif Case3v9 > log_likelihood
      classMatrix(3,9) = classMatrix(3,9) + 1;
    elseif Case3v0 > log_likelihood
      classMatrix(3,10) = classMatrix(3,10) + 1;
    else
      classMatrix(3,3) = classMatrix(3,3) + 1;
    end;
  end;
  if (DEBUG_CLASS_4)
  % Class 4
    if Case4v1 > log_likelihood
      classMatrix(4,1) = classMatrix(4,1) + 1;
    elseif Case4v2 > log_likelihood
      classMatrix(4,2) = classMatrix(4,2) + 1;
    elseif Case4v3 > log_likelihood
      classMatrix(4,3) = classMatrix(4,3) + 1;
    elseif Case4v5 > log_likelihood
      classMatrix(4,5) = classMatrix(4,5) + 1;
    elseif Case4v6 > log_likelihood
      classMatrix(4,6) = classMatrix(4,6) + 1;
    elseif Case4v7 > log_likelihood
      classMatrix(4,7) = classMatrix(4,7) + 1;
    elseif Case4v8 > log_likelihood
      classMatrix(4,8) = classMatrix(4,8) + 1;
    elseif Case4v9 > log_likelihood
      classMatrix(4,9) = classMatrix(4,9) + 1;
    elseif Case4v0 > log_likelihood
      classMatrix(4,10) = classMatrix(4,10) + 1;
    else
      classMatrix(4,4) = classMatrix(4,4) + 1;
    end;
  end;

  if (DEBUG_CLASS_5)
  % Class 5
    if Case5v1 > log_likelihood
```

```matlab
      classMatrix(5,1) = classMatrix(5,1) + 1;
    elseif Case5v2 > log_likelihood
      classMatrix(5,2) = classMatrix(5,2) + 1;
    elseif Case5v3 > log_likelihood
      classMatrix(5,3) = classMatrix(5,3) + 1;
    elseif Case5v4 > log_likelihood
      classMatrix(5,4) = classMatrix(5,4) + 1;
    elseif Case5v6 > log_likelihood
      classMatrix(5,6) = classMatrix(5,6) + 1;
    elseif Case5v7 > log_likelihood
      classMatrix(5,7) = classMatrix(5,7) + 1;
    elseif Case5v8 > log_likelihood
      classMatrix(5,8) = classMatrix(5,8) + 1;
    elseif Case5v9 > log_likelihood
      classMatrix(5,9) = classMatrix(5,9) + 1;
    elseif Case5v0 > log_likelihood
      classMatrix(5,10) = classMatrix(5,10) + 1;
    else
      classMatrix(5,5) = classMatrix(5,5) + 1;
    end;
end;

if (DEBUG_CLASS_6)
% Class 6
  if Case6v1 > log_likelihood
    classMatrix(6,1) = classMatrix(6,1) + 1;
  elseif Case6v2 > log_likelihood
    classMatrix(6,2) = classMatrix(6,2) + 1;
  elseif Case6v3 > log_likelihood
    classMatrix(6,3) = classMatrix(6,3) + 1;
  elseif Case6v4 > log_likelihood
    classMatrix(6,4) = classMatrix(6,4) + 1;
  elseif Case6v5 > log_likelihood
    classMatrix(6,5) = classMatrix(6,5) + 1;
  elseif Case6v7 > log_likelihood
    classMatrix(6,7) = classMatrix(6,7) + 1;
  elseif Case6v8 > log_likelihood
    classMatrix(6,8) = classMatrix(6,8) + 1;
  elseif Case6v9 > log_likelihood
    classMatrix(6,9) = classMatrix(6,9) + 1;
  elseif Case6v0 > log_likelihood
    classMatrix(6,10) = classMatrix(6,10) + 1;
  else
    classMatrix(6,6) = classMatrix(6,6) + 1;
  end;
end;

if (DEBUG_CLASS_7)
% Class 7
  if Case7v1 > log_likelihood
    classMatrix(7,1) = classMatrix(7,1) + 1;
  elseif Case7v2 > log_likelihood
    classMatrix(7,2) = classMatrix(7,2) + 1;
  elseif Case7v3 > log_likelihood
    classMatrix(7,3) = classMatrix(7,3) + 1;
  elseif Case7v4 > log_likelihood
    classMatrix(7,4) = classMatrix(7,4) + 1;
  elseif Case7v5 > log_likelihood
    classMatrix(7,5) = classMatrix(7,5) + 1;
  elseif Case7v6 > log_likelihood
    classMatrix(7,6) = classMatrix(7,6) + 1;
  elseif Case7v8 > log_likelihood
    classMatrix(7,8) = classMatrix(7,8) + 1;
  elseif Case7v9 > log_likelihood
    classMatrix(7,9) = classMatrix(7,9) + 1;
  elseif Case7v0 > log_likelihood
    classMatrix(7,10) = classMatrix(7,10) + 1;
  else
    classMatrix(7,7) = classMatrix(7,7) + 1;
  end;
```

```
end;

if (DEBUG_CLASS_8)
% Class 8
  if Case8v1 > log_likelihood
    classMatrix(8,1) = classMatrix(8,1) + 1;
  elseif Case8v2 > log_likelihood
    classMatrix(8,2) = classMatrix(8,2) + 1;
  elseif Case8v3 > log_likelihood
    classMatrix(8,3) = classMatrix(8,3) + 1;
  elseif Case8v4 > log_likelihood
    classMatrix(8,4) = classMatrix(8,4) + 1;
  elseif Case8v5 > log_likelihood
    classMatrix(8,5) = classMatrix(8,5) + 1;
  elseif Case8v6 > log_likelihood
    classMatrix(8,6) = classMatrix(8,6) + 1;
  elseif Case8v7 > log_likelihood
    classMatrix(8,7) = classMatrix(8,7) + 1;
  elseif Case8v9 > log_likelihood
    classMatrix(8,9) = classMatrix(8,9) + 1;
  elseif Case8v0 > log_likelihood
    classMatrix(8,10) = classMatrix(8,10) + 1;
  else
    classMatrix(8,8) = classMatrix(8,8) + 1;
  end;
end;

if (DEBUG_CLASS_9)
% Class 9
  if Case9v1 > log_likelihood
    classMatrix(9,1) = classMatrix(9,1) + 1;
  elseif Case9v2 > log_likelihood
    classMatrix(9,2) = classMatrix(9,2) + 1;
  elseif Case9v3 > log_likelihood
    classMatrix(9,3) = classMatrix(9,3) + 1;
  elseif Case9v4 > log_likelihood
    classMatrix(9,4) = classMatrix(9,4) + 1;
  elseif Case9v5 > log_likelihood
    classMatrix(9,5) = classMatrix(9,5) + 1;
  elseif Case9v6 > log_likelihood
    classMatrix(9,6) = classMatrix(9,6) + 1;
  elseif Case9v7 > log_likelihood
    classMatrix(9,7) = classMatrix(9,7) + 1;
  elseif Case9v8 > log_likelihood
    classMatrix(9,8) = classMatrix(9,8) + 1;
  elseif Case9v0 > log_likelihood
    classMatrix(9,10) = classMatrix(9,10) + 1;
  else
    classMatrix(9,9) = classMatrix(9,9) + 1;
  end;
end;

if (DEBUG_CLASS_0)
% Class 0
  if Case0v1 > log_likelihood
    classMatrix(10,1) = classMatrix(10,1) + 1;
  elseif Case0v2 > log_likelihood
    classMatrix(10,2) = classMatrix(10,2) + 1;
  elseif Case0v3 > log_likelihood
    classMatrix(10,3) = classMatrix(10,3) + 1;
  elseif Case0v4 > log_likelihood
    classMatrix(10,4) = classMatrix(10,4) + 1;
  elseif Case0v5 > log_likelihood
    classMatrix(10,5) = classMatrix(10,5) + 1;
  elseif Case0v6 > log_likelihood
    classMatrix(10,6) = classMatrix(10,6) + 1;
  elseif Case0v7 > log_likelihood
    classMatrix(10,7) = classMatrix(10,7) + 1;
  elseif Case0v8 > log_likelihood
    classMatrix(10,8) = classMatrix(10,8) + 1;
```

```
    elseif Case0v9 > log_likelihood
      classMatrix(10,9) = classMatrix(10,9) + 1;
    else
      classMatrix(10,10) = classMatrix(10,10) + 1;
    end;
end;

end  % for loop

% print results of tallie
TotalCorrect = 0;
for i = 1:1:10;
   TotalCorrect= TotalCorrect + classMatrix(i,i);
end;
Output1 = ['********** training samples = ' num2str(trainPts) '.  CD = ' num2str(NumOfCDfeat) '.  CA = '
num2str(NumOfCAfeat) '**********'];
Output2 = ['********** The total number correct = ' num2str(TotalCorrect) ' out of ' num2str(testPts * 10) '. **********'];

Output1
classMatrix
Output2
```